

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 01-191270
(43)Date of publication of application : 01.08.1989

(51)Int.Cl. G06F 15/62

(21)Application number : 63-014383 (71)Applicant : HITACHI LTD
HITACHI ENG CO LTD
(22)Date of filing : 27.01.1988 (72)Inventor : MORITA YASUSHI

(54) GRAPHIC EDITOR

(57)Abstract:

PURPOSE: To attain a high speed response by retrieving graphic information by introducing a binary tree to a system for retrieving graphic information by defining a coordinate to be a key and editing and outputting.

CONSTITUTION: In order to retrieve the large quantity of the graphic information, the binary tree for enclosing the graphic information, holding a rectangle coordinate (the diagonal vertex of a rectangle, for instance, coordinates of lower left and upper right vertexes of the rectangle) circumscribing thereto and constituted of plural nodes (respectively corresponding to one graphic a piece of graphic information) having a priority applied respectively by defining the coordinate to be a reference is introduced to retrieve the graphic information based on this binary tree. At the time of retrieving, all the tree nodes are not searched but the retrieval according to the priority of the node is carried out. Thereby, the target graphic information can be made access at high speed.

LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's
decision of rejection]

[Date of extinction of right]

⑫ 公開特許公報(A) 平1-191270

⑤Int.Cl.⁴

識別記号

庁内整理番号

⑬公開 平成1年(1989)8月1日

G 06 F 15/62

3 2 0

K-6615-5B

審査請求 未請求 請求項の数 6 (全17頁)

⑭発明の名称 図形編集装置

⑯特 願 昭63-14383

⑰出 願 昭63(1988)1月27日

⑱発明者 森 田 靖 茨城県日立市幸町3丁目2番1号 日立エンジニアリング株式会社内
 ⑲出願人 株式会社日立製作所 東京都千代田区神田駿河台4丁目6番地
 ⑲出願人 日立エンジニアリング株式会社 茨城県日立市幸町3丁目2番1号
 ⑳代理人 弁理士 平木 道人

明 細 書

1. 発明の名称

図形編集装置

2. 特許請求の範囲

(1) 図形情報を入力し、記憶する手段と、前記図形情報を指定された位置に表示する手段と、前記図形情報を囲み、これに外接する矩形の1対の対角頂点の座標を各図形毎に演算し、記憶する手段と、前記各図形毎の矩形の1対の対角頂点の座標に基づいてバイナリ・ツリーを生成する手段とを具備したことを特徴とする図形編集装置。

(2) 前記バイナリ・ツリーは、ルート・ノードおよび他のツリー・ノードの集合よりなり、前記ルート・ノードおよび他のツリー・ノードはその下に接続される右および左の子ノードを有することを特徴とする前記特許請求の範囲第1項記載の図形編集装置。

(3) 前記ルート・ノードは、空ノードであるダミーノードに接続されることを特徴とする前記特許請求の範囲第2項記載の図形編集装置。

(4) 前記バイナリ・ツリーの各ノードは、あるノード n に対応する矩形の1対の対角頂点座標を $k_1(n)$ 、 $k_2(n)$ (ただし、 $k_1(n) \leq k_2(n)$)、当該ノード n に対応した区間を (α, β) とし、さらに当該ノードを a 、その左ノードを b 、またその右ノードを c としたとき、ノード a 、 b 、 c が次の条件を満足することを特徴とする前記特許請求の範囲第1ないし第3項のいずれかに記載の図形編集装置。

イ) ノード a 、

$$k_1(a) \leq k_1(b), k_1(a) \leq k_1(c), \\ \alpha \leq k_2(a) \leq \beta,$$

ロ) ノード b 、

$$\alpha \leq k_2(b) \leq (\alpha + \beta) / 2,$$

ハ) ノード c 、

$$(\alpha + \beta) / 2 < k_2(c) \leq \beta,$$

(5) 前記バイナリ・ツリーは、図形情報とともに記憶されることを特徴とする前記特許請求の範囲第1ないし第4項のいずれかに記載の図形編集装置。

(6) 前記バイナリ・ツリーは、図形情報とともに記憶されず、編集操作開始時に生成されることを特徴とする前記特許請求の範囲第1ないし第5項のいずれかに記載の図形編集装置。

3. 発明の詳細な説明

(産業上の利用分野)

本発明は、グラフィックス・システムにおける一般的な図形編集装置に係り、特に大量の図形情報の拡大・縮小表示出力や、各図形の拡大・縮小・移動・複写・削除等を高速に行うのに好適な図形編集装置に関する。

(従来の技術)

各省庁・官庁等をはじめとして一般企業においても、住宅地図等の大量の図形情報を、原図等に1つ1つ書きによって記入修正したり、保存したりする従来の方法に代って、グラフィック・ディスプレイを用いて作画、編集を行い、ブロックやLBP(レーザー・ビーム・プリンタ)等を用いて出図する、図形編集用電子計算機システムが導入されつつある。

ピックされた図形情報は、ブリンク等を施してディスプレイに再表示され、各編集の可否を操作者に問い合わせる。例えば、移動の実行が操作者により指示された場合は、ブリンクされた図形を管面色(地色)等で上書き描画することによって当該図形を消去する。

消去図形が管面色以外の図形色部分に重なっている場合には、前記消去によって、その部分に歯抜け部を生ずる。この歯抜け部を修復するためには、消去後に、例えば特開昭60-173677号公報に示されるように、消去された図形により管面色にて上書きされた図形情報を、全図形情報より検索し、前記消去図形の一部を前記歯抜け部のみに、当該部分の色で再表示する。このようにして、歯抜けとなった表示画面を修復した後、ピックされた図形を移動先に表示すれば、移動操作を完了する。

(発明が解決しようとする課題)

しかしながら、かかる従来の図形編集方式においては、次のような問題が発生する。

このようなグラフィックス・システム(図形編集方式)を実現するために、グラフィック・ディスプレイをはじめとする各ハードウェアは、図形情報が出力可能範囲(例えば、グラフィック・ディスプレイであれば、1画面分に相当するフレーム・バッファの容量)を超えた場合には、出力図形情報をクリッピングしてその範囲内に収まるように出力する機能を有している。

例えば、1管(画)面に描画した図形情報を拡大して表示し、画面上で図形情報を編集するような場合、全図形情報に対して拡大座標変換を行って出力したとしても、1管面分にクリッピングして1画面上に正しく(誤動作なしに)表示できるようになっている。

また、グラフィック・ディスプレイを用いて、会話形式で図形情報の移動等の編集操作を実現するために、例えば特開昭62-57078号公報に示すような方式を用い、マウス等で指示された座標をもとに、全図形情報を検索して編集対象の図形情報をピックする方式がある。

1点目は、図形情報を表示、出図する際に発生するもので、大量の図形情報の一部しか表示、出図の対象とはならない場合でも、全図形情報を処理した後に出力しなければならないために、表示、出力時間が非常に遅くなることである。

2点目は、図形情報を会話形式で編集する際に発生するもので、図形のピック、消去、移動後の修復のために、全図形情報を検索する必要があるために、検索時間が長くなって応答性が悪く、誤操作の原因ともなることである。

3点目は、上記の各問題点の故に、応答性を確保するためには、大量の図形情報を一元的に管理したり、編集したりすることができないので、少量(適当量)の図形情報ファイルに分割して、個々に編集せざるを得ず、このため、特に分割境界にまたがるような図形情報などは、複数の図形情報ファイル等に格納されるため、該図形情報を変更する場合は、複数の図形情報として編集を複数回行うことが必要となることである。

4点目も、3点目と同様の理由により、分割し

た図形情報単位の表示、出力しかできないために、これらを統合して連続した表示、出力を行うことは、非常に難しい。

このように、従来の図形編集方式では、大量の図形情報を一括して取扱おうとすれば、膨大な時間がかかり、応答性に劣るため、操作者に対しては極めて不便になる。

一方、応答性を確保するために、図形情報を分割して取扱えば、図形情報の編集時には極めて複雑な作業を強いられ、視認性の良い図形情報の出力ができないという問題があった。

本発明の目的は、このような従来方式の問題点を解決し、操作者に対する応答性と、情報の一元管理・編集及び視認性の良い図形情報の出力などが全て満足される図形編集方式を提供することにある。

(課題を解決するための手段)

本発明では、膨大な図形情報を検索するために、図形情報を囲み、それに外接する矩形座標(矩形の対角頂点、例えば矩形の左下、右上頂点の座標)

を保持し、その座標を基準として各々優先順位を付けた複数のノード(それぞれが1つの図形情報に対応)によって構成されたバイナリ・ツリーを導入し、このバイナリ・ツリーをもとに図形情報を検索する。

検索に際しては、全てのツリー・ノードを探索するのではなく、ノードの優先順に従った探索を行う。優先順位を示すキーとしては、図形情報を囲む矩形座標のうち、x座標のペア、またはy座標のペアを用いる。

さらに各ノードには、図形情報の存在する区間座標が対応している。以下に、その構造を定義する。

$k_1(n), k_2(n)$:

ノードnのx座標、またはy座標のペア。

ただし、 $k_1(n) \leq k_2(n)$

α, β : ノードに対応した区間

a : ノード

b : ノードaの左ノード

c : ノードaの右ノード

上記のような定義の下では、本発明のバイナリ・ツリーの各ノードは以下の条件を満足する。

1) ノードaに対して、

$$k_1(a) \leq k_1(b), k_1(a) \leq k_1(c)$$

$$\alpha \leq k_2(a) \leq \beta$$

2) ノードbに対して、

$$\alpha \leq k_2(b) \leq (\alpha + \beta) / 2$$

3) ノードcに対し

$$(\alpha + \beta) / 2 < k_2(c) \leq \beta$$

上記の条件を、本ツリーの全てのノードに対して再帰的に適用することにより、バイナリ・ツリーが構成できるものである。

本構造に従って、ノードの探索を行い、検索する必要のないサブツリーをチェックすることにより、不要な検索を行わず、目的の図形情報を高速に確定し得るものである。

(作用)

図形情報の検索に際しては、前述のバイナリ・ツリー構造を用い、不必要な情報を検索しないこ

とにより、目的の図形情報を高速にアクセスできる。

(実施例)

以下、本発明の1実施例を詳細に説明する。

第1図は、本発明による図形情報の構成を示すもので、同図(A)はバイナリ・ツリーの1例を示す図、同図(B)は前記バイナリ・ツリーで表わされる具体的な図形情報群の1例を示す図である。第2図は、本発明を実施するシステム構成例である。

はじめに、本発明による会話形図形編集装置の概略構成について、第2図を参照して述べる。

本発明の図形編集装置への図形情報の入力は、グラフィック・ディスプレイ11やキーボード12等を使用して、電子計算機13に格納された会話形式図形編集プログラムを起動し、種々の情報の入力は、プログラムと会話しながら、マウス14、ライトペン15、デジタイザ16、キーボード12等の各種入力機器(座標入力手段)を用

いて行う。20はバッチ型入力装置である。

起動されたプログラムは、操作者の指示にしたがった図形情報（例えば、日立市大みか町1丁目の住宅地図）を、グラフィック・ディスプレイ11に表示する。

表示の際は、2次記憶装置17に格納された全図形情報（日立市全住宅地図）を読み出し、指示のあった図形情報（大みか1丁目）を検索し、適当な座標変換（拡大・縮小及びハードウェアの座標系への変換）を施し、グラフィック・ディスプレイ11に出力する。

操作者は目的の画面が表示されると、本プログラムと会話形式で、各種の入力機器を用いて指示を与えることにより、図形情報の編集（新しい住宅の追加及び、取り壊された住宅の削除等）を行う。

例えば、新たな線分を描画する場合には、マウス14を用いて、グラフィック・ディスプレイ11上で、所望の始点および終点をクリックすると、所望の直線が追加表示される。また、既に表

示されている線分の近傍をクリックすることによって、その線分をピックアップし、前記線分を削除したり、あるいは、さらに他の位置をクリックすることによって、その線分を移動したりする。

プログラムは、各編集操作に従って、図形をグラフィック・ディスプレイ11に表示したり、図形情報の更新を行ったりする。操作者より編集終了の指示があった場合、プログラムは更新された図形情報を2次記憶装置17に格納して停止する。

また、2次記憶装置17に格納した情報の中から、操作者の指示に従って、（例えば日立市大みか1丁目の住宅地図）を、LBP18やブロック19に選択的に出力する機能も有している。

続いて、本システムの図形情報の構成について述べる。

第1図(A)は本発明によるバイナリ・ツリー（以下、ツリーと略する）1の構造例を示し、同図(B)はこのツリーで表わされる図形情報2の1例を図示したものである。

ここで、ツリーの各ノード3はそれぞれ、個々の図形情報4に対応付けられている（ノード3に含まれる情報については、後述する）。その対応関係は、各ノードの左（または右）上に記された丸枠付数字5と、個々の図形情報の左上に記された丸枠付数字6によって示されている。

ツリーの構造は、個々の図形情報に外接する矩形7の1対角線の左下、右上のx座標（以下、 $x_1(i)$ 、 $x_2(i)$ ： $i=1, 2, 3 \dots$ 等と略する）と、各ノードに対応付けられたx座標の区間8に従っている。なおここでは、説明を簡単にするため、全図形情報の持つ座標空間は、第1図(B)に示したように、 $[0, N] / [0, M]$ の整数座標に正規化されているものとする。

ツリーのルート・ノード9は、区間 $[0, N]$ に右上座標 $x_2(1)$ が含まれている複数の図形情報（すなわち、第1図(B)に示された各図形）の中で最小の左下座標 $x_1(1)$ を持つ図形情報（第1図の例では、折線図形）が対応する。

ルート・ノード9の左ノード（バイナリ・

ツリーで左側へ分岐した子ノード）は、区間 $[0, N/2]$ に $x_2(j)$ （ただし $j \neq i$ が含まれているものの中で、最小の $x_1(j)$ （ただし、 $j \neq i$ ）を持つ図形情報（第1図の例では、菱形図形）が対応し、また右ノード（バイナリ・ツリーで右側へ分岐した子ノード）は、区間 $[N/2+1, N]$ に $x_2(k)$ （ただし $k \neq i$ ）が含まれている中で最小の $x_1(k)$ （ただし、 $k \neq i$ ）を持つ図形情報（第1図の例では、矩形図形）が対応する。

以上の説明から分るように、本ツリー構造の一般的定義はつぎのようになる。

- (1) あるサブツリーのルート・ノードは、区間 $[\alpha, \beta]$ に $x_2(1)$ が含まれており、最小の $x_1(1)$ を持つ図形情報が対応する。
- (2) 前記ルート・ノードの左ノードは、区間 $[\alpha, (\alpha + \beta) / 2]$ に $x_2(j)$ （ $j \neq i$ ）が含まれているものの中で、最小の $x_1(j)$ （ $j \neq i$ ）を持つ図形情報が対応する。
- (3) その右ノードは、区間 $[(\alpha + \beta) / 2 + 1,$

β] に $x_2(k)$ ($k \neq i$) が含まれているものの中で、最小の $x_1(k)$ ($k \neq i$) を持つ図形情報が対応する。

上記の定義を全てのサブツリーが満足するように、第1図(B)の全図形に順次繰り返し適用して構成したものが、第1図(A)のバイナリ・ツリーである。なお、ルート・ノードを子ノードとして持つノード10は、ツリー操作(例えば、第1図において、ルートノード9を削除するような操作を容易にするためのダミーノードである。

ツリーの各ノードは、第1図(A)にも示したように、当該図形情報を保持するためのポイントと、当該図形を囲んで外接する矩形の左上、右下の座標、及び図形情報へのポイント(例えば、メモリの先頭アドレス、データ長)、左または右の子ノードへのポイント等を保持している。また、出力結果として、図形が描画順序に依存するような場合は、ノードを描画順に連結した双方向のポイント等をも保持することができる。

次に、本発明の全体的な動作を説明する前に、

れた前記座標 X_1, Y_1, X_2, Y_2 を基準とし、各図形情報を囲む矩形領域の座標をこれと比較して行う。また、ツリートラバースは、当該ツリーのルート・ノードより行なう。たどったノード(現在検索中のノード)を n (401) とし、ノードに対応した区間 $[\alpha, \beta]$ を $[0, N]$ とする(402)。

つぎに、検索中のノード n が空ノードであるか、換言すれば、次にたどるべき子ノードを持っているか否かをチェックする(403)。空ノードならば、410の判定を行ない、空ノードでなければ404の判定を行なう。

ここで、検索対象ノードに対応した図形情報を囲む矩形領域の左下および右上隅の座標を、第1図(B)のように、 $\{x_1(n), y_1(n)\}$ 、 $\{x_2(n), y_2(n)\}$ とし、さらに $x_1(n) \leq x_2(n)$ 、 $y_1(n) \leq y_2(n)$ とする。

また、ツリートラバースは、以下の説明から明らかなように、深さ優先で行なう。換言すれば、403の判定が否定で、検索対象ノードに子ノ

まず、動作の中心となるバイナリ・ツリーを用いた図形情報の検索、図形情報の追加、削除の手順につき、ツリートラバースとノード操作を重点にして説明する。なお、これらの操作手順を、第1図の具体的なバイナリ・ツリーに適用する例については、後で説明する。

まず、第4図に示すフローチャートを参照して、バイナリ・ツリーの構造に従った図形情報の検索について述べる。

グラフィック・ディスプレイに表示すべき範囲を、第13図に示したように、領域30 $[0, N] \times [0, M]$ 上(例えば、日上市全体の地図)の $[X_1, Y_1] \times [X_2, Y_2]$ の範囲32(例えば、大みか1丁目のが含まれる範囲)とする。即ち、表示しようとしている図形情報の全て、または一部が $[X_1, Y_1] \times [X_2, Y_2]$ の範囲にあるもの(第13図において、符号34, 36であらわされたもの)を、全図形情報の中から検索するものとする。

検索時のツリートラバース(探索)は、与えら

ドがある限りツリーを下方へたどり、たどった順にノードおよび区間をスタック(後入れ先出し)方式で記憶する。

バイナリ・ツリーの構成から明らかなように、現在の検索対象ノードの持つ矩形の左下 x 座標 $x_1(n)$ が出力範囲の右側(例えば、第13図の矩形38)であれば $(X_2 < x_1(n))$ 、当該ノード以下のサブツリーは、全て出力範囲外なので、検索しない(404の前半の条件)。

また、現在のノードに対応する区間 $[\alpha, \beta]$ の終点 (β) が出力範囲の左側 $(\beta < X_1)$ であれば、それ以下のサブツリーは全て左側、すなわち範囲外となるので検索しない(404の後半の条件)。

前記の判定404で範囲外とされたノード以外のものについては、そのノードの持つ矩形が出力範囲 $[X_1, Y_1] \times [X_2, Y_2]$ に含まれるかどうかを判定する(405)。すなわち、405の判定が不成立ならば範囲外であり、一方、同判定が成立するノードは範囲内であるので、出

力の対象として保持する(406)。

続いて、現在の検索対象ノードとその区間 $[\alpha, \beta]$ をスタック方式でプッシュ(入力)し(407)、 $[\alpha, \beta]$ の区間計算を行なって、その左のノードにトラバースし(408, 409)、判定403へ戻る。

このとき、もし403の判定が成立し、ノードが無く空ノードであると判定されれば、410でスタックが空か否かを判定する。スタックが空でないと判定されたときは(410)、スタックをポップし(411)、右のノードにトラバースし(412, 413)、判定403へ戻り、404以降の判定、操作を繰り返す。判定403の結果が肯定でノードが無く(空ノードで)、しかも判定410の結果も肯定でスタックも空であるならば、全ての所望ノードについての検索が終了したことになるので、処理を停止する。

続いて、第5図に示すフローチャートを参照して、ツリー・ノードの生成、追加処理について述べる。 n, α, β, x, y 等は前記検索処理の場

すべきノードとする(510)。

更に、挿入すべきノードの矩形の右上 x 座標 x_2 (new)が $[\alpha, (\alpha + \beta) / 2]$ に含まれれば、左のサブツリーに挿入され(512, 513~515)、そうでなければ右のサブツリーに挿入される(512, 516~518)。

最後に、挿入すべきノードは、必ずツリーの葉として生成される(519)ので、その左右の子ノードを空ノードとして(520)処理を終了する。

ツリー・ノード操作処理の他の例として、ノードの削除処理について、第6図のフローチャートを参照して説明する。 $n, \alpha, \beta, x_1, y_1$ 等は、前記仮定と同様である。また、ダミーのノードより処理を開始することも、前述と同様である(602~605)。

削除すべきノードを前述の検索処理によって検索、確定し、これを del とする(601)。 del とされたノードは本ツリーは、前に詳述した定義と手法に従って構成されているため、

合の仮定と同様である。またノード生成動作をより容易に実現するために、ここではルート・ノードを左に持つダミーのノードより処理を開始するものとする。

新しく挿入すべきノードを new とし(501)、ツリーの構造にしたがって各ノードをたどる。すなわち、 P_n をダミーノード(第1図の10)とし(502)、 P_n の左ノードを n にセットする(503)。 in に「左」を示すフラグ値をセットし(504)、区間 (α, β) に $(0, N)$ をセットする(505)。

つぎの判定506では、 n が空ノードか否かをチェックし、空ノードでなければ判定507を行なう。この判定507において、現在の検索対象ノードの矩形の左下 x 座標 $x_1(n)$ が、挿入すべきノードの矩形の左下 x 座標 $x_1(new)$ よりも大きければ、挿入すべき位置は、現ノードの位置であるので、現在のノードと挿入すべきノードの各ポイントを付け替えることにより、両ノードを、交換し(508, 509)、現在のノードを挿入

ノード生成時と同様の手法で、その構造に従ってツリーをたどることにより(606~614)、 del の位置(del の親ノード P_n 、左右区別 d_n)を探索することができる。

del が発見された(606)ならば、 del をツリーより削除する。すなわち、ツリーの構造を保つために、下記のようにして、 del の子(左, 右)ノードによりそのノードを置き換える。

del の右ノードが無い(空ノード)か(616)、または、 del の左ノードの矩形左下 x 座標が、右ノードの矩形左下 x 座標よりも小さい場合(618)は、 del の左ノードで置き換える。この場合の置き換えは、 del の親ノードに del の左ノードを接続(626)し、 del の右ノードを左ノードの右ノードとし(625, 630)、 del に del の左ノードのポイントを保持させること(629)によって行なう。

また、 del の右ノードで置き換える場合(617, 618)も同様に行なう(619~

624)。

このようにして、削除すべきノードを葉に向けて一段一段置き換えてゆき(615)、葉ノードを削除したところで(631)、この処理は終了する。

つぎに、第1図に示す図形情報をもとに、前述の各編集操作、すなわち、図形情報の新規生成、追加及び削除の各操作を通しての、ツリーの状態の経緯を具体的に述べる。

まず、図形情報を新たに生成する場合についての例を、第7図および第5図を参照して述べる。

新規に図形情報を生成する場合は、まず、ダミーのツリー・ノード10を生成しておく(701)。

その後、第1図②の図形情報が操作者により入力(例えば、マウス14等で多角形の各頂点をグラフィック・ディスプレイ上でクリックすることにより)されたと仮定すると、その図形情報より外接矩形座標 $\{x_1(2), y_1(2)\}$, $\{x_2(2), y_2(2)\}$ を求めて、新たなノード3に格納する。

成した後、新たなノードに格納する。この後、第5図に示すフローチャートと第8図に示す経緯図に従い、③を格納したノードをnewとし(501)、Pnを第1図のダミーノード10として(502)、ツリーの生成を行なう(802)。

この場合は、Pnの左ノードにはルート・ノード(②のノード)がセットされているため、nには②のノードがセットされる(503)。504~505の処理後、nすなわちノード②は空ノードではない(506)ので、 $x_1(\text{new}) (=x_1(3))$ と $x_1(n) (=x_1(2))$ とを比較する(507)。

$x_1(\text{new}) < x_1(n)$ ではないので、Pnにn(=②のノード)をセットし(511)、その後 $x_2(\text{new})$ と $(\alpha + \beta) / 2 (=N / 2)$ を比較する(512)。 $x_2(\text{new}) \leq (\alpha + \beta) / 2$ であるので、nの左のノード(空ノード)をnとし(513)、 $[\alpha, \beta]$ に $[\alpha, (\alpha + \beta) / 2]$ (= $[0, N / 2]$) とセットして(514)、

この後、第5図に示すフローチャートに従い、格納したノードをnew(=②のノード)とし(501)、Pnを第1図のダミーノード10として(502)、ツリーの生成を行なう(702)。

この場合は、Pnを(ダミーノード)の左、右のノードは存在せず、空ノードとなっているため、nには空ノードがセットされ(503)、504~506の処理後、直ちにノード挿入処理、Pn(=ダミーノード)のin(=左)ノードにnew(=②のノード)をセットし(519)(703)、newの左、右ノードを“空ノード”にセットする(520)(704)を行なう。以上で、図形情報②の新規生成を行なう手順は終了する。

次に、前記状態より第1図に示した③、⑤、①の図形情報がこの順に入力され、図形情報を追加する場合について、各々第8、9、10図を参照して説明する。

③の図形情報が入力されて、外接矩形座標を生

inに左を示すフラグ値をセット(515)する(803)。

続いて、nには空ノードがセットされていることが判る(506)ので、ノード挿入処理(519, 520)を行なう(804, 805)。

ノード挿入後は、描画順を保持するため、②を格納したノードに、③のノード・ポインタ(②の次に③が描画されていることを示す。)をセットし、一方、③を格納したノードには、②のノード・ポインタ(③の前に②が描画されていることを示す。)をセットする(806)。

これにより、図形情報を描画順に表示することが可能となり、図形情報が重なるような位置に入力され、描画されても、入力した通りに表示することが可能となる。例えば、赤色に塗りつぶした長方形の上に、白色で文字A, B, C等と描画し、これらの文字を強調する等のことが可能になる。

続いて、⑤の図形情報を入力した場合のノードの挿入は、③を入力した場合と同様に、第5図に示すフローチャートと第9図に示す経緯図に従っ

て行なわれる。

すなわち、⑤のノードをnewとして(501)、ダミーのノードよりツリー生成を開始する(502~505)(901, 902)。nは②のノードであり、判定506は不成立であるので、 $x_1(\text{new}) (=x_1(5))$ と $x_1(n) (=x_1(2))$ とを比較する(507)。 $x_1(\text{new}) < x_1(n)$ ではないので、Pnにnをセットし(511)、 $x_2(\text{new}) (=x_2(5))$ と $(\alpha + \beta) / 2 (=N / 2)$ を比較する(512)。

$x_2(\text{new}) > (\alpha + \beta) / 2$ 、すなわち前記判定512は否定であるので、nの右ノード(空ノード)にnをセットし(516)、 $[\alpha, \beta]$ に $[(\alpha + \beta) / 2 + 1, \beta]$ ($= [N / 2 + 1, N]$)とセットして(517)、inに右を示すフラグ値をセット(518)する(903)。

続いて、nに空ノードがセットされていることが判る(506)ので、ノード挿入処理(519, 520)を行なう(904, 905)。ノード挿

入後は、描画順保持のためのポイントを生成する。

即ち、③のノードに、⑤のノード・ポイントを格納する。その結果、第9図に906で示したように、③には、③の前の図形情報を示す②のノード・ポイントと、③の次の図形情報を示す⑤のノード・ポイントの2つが保持される。また、⑤のノードには、③のノード・ポイントをセットする(906)。

①の図形情報に対するノードの挿入は、前述した図形情報③、⑤の場合と同様に、第5図に示すフローチャートと第10図に示す経緯図に従って行なわれる。

①のノードをnewとして(501)、ダミーのノードよりツリー生成を開始する(502~505)(1001, 1002)。ここで、nは②のノードであるから空ノードではなく(506)、 $x_1(\text{new}) (=x_1(1)) < x_1(n) (=x_1(2))$ ので(507)、newの左、右ノードに各々nの左ノード(③のノード)、右ノード(⑤のノード)をセットする(508)(1003)。

この段階では、newの左ノードは③のノード、右ノードには、⑤のノードがセットされている。

次に、Pn (=ダミーノード)のin (=左)ノードにnew (=①のノード)をセットする(509)(1004)。そして、nとnewとを交換する(510)(1005)。この場合、nは①のノードとなり、newは②のノードとなる。

つまり、ツリーの途中の位置にノードが挿入された場合は、挿入すべきノードを交換して、③や⑤のノード挿入時と同様にツリーをたどる。

即ち、Pnにn (=①のノード)をセットし、 $x_2(\text{new}) (=x_2(2)) \leq (\alpha + \beta) / 2 (=N / 2)$ であるので(512)、nにnの左ノード(③のノード)、 $[\alpha, \beta]$ に $[\alpha, (\alpha + \beta) / 2]$ ($= [0, N / 2]$)、inに左を示すフラグ値をセットする(513~515)(1006)。

更に、同様にして、n (=③のノード)の判定を行なう(506)。 $x_1(\text{new}) (=x_1(2)) < x_1(n) (=x_1(3))$ であるから(507)、new (=②のノード)の左、右ノードにnの左、

右ノード(ともに空ノード)を各々セットし(508)(1007)、Pn (=①のノード)のin (=左)ノードをnewとセットして(509)、nとnewの交換をする(510)(1008)。ここでnew = ③のノード、n = ②のノードとなっている。

続いて、Pnにn (=②のノード)をセットする(511)(1009)。 $x_2(\text{new}) (=x_2(3)) \leq (\alpha + \beta) / 2 (=N / 4)$ であるため(512)、nにnの左ノード(空ノード)を、 $[\alpha, \beta]$ に $[\alpha, (\alpha + \beta) / 2]$ ($= [0, N / 4]$)をinに左を示すフラグ値をセットする(513~515)(1010)。

nは空ノードであるので(506)、Pn (=②のノード)のin (=左)ノードに、new (=③のノード)をセットし(519)(1011)、newの左右ノードを空ノードにセットする(520)(1012)。

上述のツリー生成処理後は、描画順ポイントを、③、⑤の入力時と同様の手順で、①のノードと、

⑤のノードにセットする(1 0 1 3)。

以上に詳細に述べたように、本発明では、ツリーの生成時には、ノードごとに2分された枝を、1つのパスしか通らない。即ち、1つのノード挿入のためには、ツリーのルートより葉までの一本道しかたどらないため、たどるノードの数は、全ノード数(すなわち、全図形情報数)をKとした場合、ほぼ $1 \circ g_2 K$ でよいことになる。

従って、全図形情報が100000程度の多数である場合でも、17回程度($\approx 1 \circ g_2 100000$)のノード探索で操作を完了することが可能となっている。

具体例の最後として、図形情報を削除する場合について、以下に説明する。

図形情報を削除する場合は、操作者が入力した座標(例えば、マウス等で削除対象となる図形の近傍を、クリックした位置)をもとに、第4図で示されるツリー探索処理を用いて、対象図形情報に対応したツリー・ノードを確定し、第6図で示されるツリー・ノード削除処理にて、ノードを削

除し、更に図形情報も削除する。

第1図に示されたツリー及び図形情報が記憶されており、②の図形情報のひし形の右下辺上の座標が、削除対象として与えられた場合に、②の図形情報を削除するまでの経緯についての例を、以下に述べる。

まず入力された座標を (x, y) とし、 $N/4 < x \leq N/2$ であり、かつ $x_1(4) < x < x_2(4)$ であると仮定する。更に、 ε をNに比較して十分小さい数として(例えば、 $N = 1024$ とすれば $\varepsilon = 8$ 程度)、

$$X_1 = x - \varepsilon,$$

$$Y_1 = y - \varepsilon,$$

$$X_2 = x + \varepsilon,$$

$$Y_2 = y + \varepsilon,$$

とする。

これらの条件をもとに、まず第4図に示すフローチャートに従い、第1図で示されるツリーをたどる例について示す。更に、どのようにツリーがたどられるかの軌跡を、第11図に示す。

第11図において、2重丸のノード①、②、④は、図形情報が削除対象か否かを検索したノード、1重丸のノード③、⑤は、ツリートラバースの過程で探索したノードである。また、三角棒のノード⑥、⑦は、探索しないノードである。また、1108は、ノードをたどる順を表す矢線である。

まず、第4図のフローチャートにおいて、nをルート・ノード(①のノード)とし(401)、 $[\alpha, \beta]$ を $[0, N]$ とする(402)。nはルート・ノードであって空ノードではなく(403)、しかも $X_2 > x_1(n)$ ($=x_1(1)$)かつ $X_1 < \beta$ ($=N$)であるので(404)、 $[X_1, Y_1] \times [X_2, Y_2]$ と $[x_1(n), y_1(n)] \times [x_2(n), y_2(n)]$ ($n=1$)との領域が重なるか否かの判定を行なう(405)。

これらの領域が重ならないので、スタックにn($=1$)、 $[\alpha, \beta]$ ($= [0, N]$)をプッシュして保持し(407)、nにn($=①$ のノード)の左ノード(②のノード)をセットする(408)。そして、次のノードに対応する区間をセットする

ために、 $[\alpha, \beta]$ に $[\alpha, (\alpha + \beta)/2]$ ($= [0, N/2]$)をセットして(409)、判定403へ戻る。

n($=②$ のノード)は空ノードではなく(403)、 $X_2 > x_1(n)$ ($=x_1(2)$)かつ $X_1 < \beta$ ($=N/2$)であり(404)、 $[X_1, Y_1] \times [X_2, Y_2]$ と $[x_1(n), y_1(n)] \times [x_2(n), y_2(n)]$ ($n=2$)との領域が重なる(405)ので、nで示されるノードは、検索されたノードである(406)。

更に同様にしてn、 $[\alpha, \beta]$ をスタックにプッシュして保持し(407)、nにnの左ノード(③のノード)とにセットし(408)、 $[\alpha, \beta]$ に $[\alpha, (\alpha + \beta)/2]$ ($= [0, N/4]$)とした後、再び判定403へ戻る。

n($=③$ のノード)は空ノードではなく(403)、 $X_1 > \beta$ ($=N/4$)であり(404)、またスタックは空でない(410)ので、スタックをポップしてノード、区間を取り出しn、 $[\alpha, \beta]$ にセットする(411)。この

時の n は②のノード、 $[\alpha, \beta]$ は $[0, N/2]$ である。

n に n の右ノード (④のノード) をセットし (412)、 $[\alpha, \beta]$ に $[(\alpha + \beta) / 2 + 1, \beta]$ ($= [N/4 + 1, N/2]$) をセットする (413)。

更に、判定403へ戻ってツリーをたどる。このときも n ($=$ ④のノード) は空ノードではなく (403)、 $X_2 > x_1(n)$ ($= x_1(4)$) かつ $X_1 < \beta$ ($= N/2$) であり (404)、 $[X_1, Y_1] \times [X_2, Y_2]$ と $[x_1(n), y_1(n)] \times [x_2(n), y_2(n)]$ ($n = 4$) とは領域が重ならない (405)。

それ故に、スタックに $n, [\alpha, \beta]$ をプッシュして保持し (407)、 n に n の左ノード (空ノード) をセットし (408)、 $[\alpha, \beta]$ に $[\alpha, (\alpha + \beta) / 2]$ ($= [N/4 + 1, (N/4 + 1 + N/2) / 2]$) をセットして (409)、判定403へ戻る。

次の判定では、 n の空ノードであり (403)、

n ($=$ ⑤ノード) は空ノードでなく (403)、 $X_2 < x_1(n)$ ($= x_1(5)$) であり (404)、スタックは空 (410) であるので、ツリー探索を終了する。

以上の処理によって、削除対象の図形情報は、②のノードに対応したものであることが分る。

ここで、実際に検索されたノードは、下記の通りとなる。

- ①, ②, ④: 対象の図形情報か否かのチェックを行なったノード。
- ③, ⑤: ツリートラバースの過程として検索したノード。

これより、対象の図形情報か否かの検索の為に、ノードごとに2分された枝の、1つのパスしか通らないことが分る。即ち、ツリー生成時と同様に、その検索対象のノード数は、全ノード数 (全図形情報数) を K とした場合 $\log_2 K$ 程度であることが分る。

従って、従来技術では、全図形情報が、たとえば100000個あった場合は、同数の

スタックは空でない (410)、スタックをポップしてノード、区間を取り出し、 $n, [\alpha, \beta]$ にセットする (411)。この時の n は④のノードであり、 $[\alpha, \beta]$ は、 $[N/4 + 1, (N/4 + 1 + N/2) / 2]$ である。

n に n の右ノード (空ノード) をセットし (412)、 $[\alpha, \beta]$ に $[(\alpha + \beta) / 2, \beta]$ ($= [N/4 + 1 + (N/4 + 1 + N/2) / 2, (N/4 + 1 + N/2) / 2]$) をセットして (413)、判定403へ戻る。

この場合も、 n は空ノードであり (403)、スタックは空でない (410)、更にスタックをポップしてノード、区間を取り出し、 $n, [\alpha, \beta]$ にセットする (411)。この時の n は①のノード、 $[\alpha, \beta]$ は $[0, N]$ である。

n を n の右ノード (⑤ノード) とし (412)、 $[\alpha, \beta]$ を $[(\alpha + \beta) / 2 + 1, \beta]$ ($= [N/2, N]$) とする (413)。その後、判定403へ戻る。

100000回図形情報を検索する必要があったが、本方式では、わずか17回程度 ($\log_2 100000$) で済む。これに、ツリートラバースのオーバーヘッド、即ち、ツリートラバースの過程として検索したノードの個数を加えても、その検索回数には、大きな開きがあることが判る。

次に、削除対象となった図形情報の②のノードを、ツリーから削除する具体例について、第6図に示されたフローチャートと第12図の経緯図を参照して説明する。

まず、 del を②のノードにセットする (601) (1201)。 P_n を第1図のダミーノード10とし (602)、 n に P_n の左ノード (ルート・ノードである①のノード) をセットし (603)、 dn に「左」を示すフラグ値をセットし (604)、区間 $[\alpha, \beta]$ に $[0, N]$ をセットする (605) (1202)。

つぎに、 n ($=$ ①のノード) は del ($=$ ②のノード) ではないので (606)、 P_n に

n (=①のノード) をセットする (607)。
 $x_2(\text{del}) (=x_2(2)) \leq (\alpha + \beta) / 2$
 (= $N/2$) であるので判定608は肯定であり、
 n に n (=①のノード) の左ノード (=②のノード) をセットする (609)。

$[\alpha, \beta]$ に $[\alpha, (\alpha + \beta) / 2]$ (= $[0, N/2]$) をセットし (610)、 dn に「左」を示すフラグ値をセットする (611)(1203)。

このとき、 n (=②のノード) は del であり (606)、 del の左、右ノードはともに空ノードでない (616~617) ので、 $x_1(\text{del}$ の左ノード) (= $x_1(3)$) と $x_1(\text{del}$ の右ノード) (= $x_1(4)$) とを比較する判定618を実施する。

ここでは、 $x_1(\text{del}$ の左ノード) < $x_1(\text{del}$ の右ノード) であり、判定618が成立するので、 del の右ノード (④のノード) を n にセットし (625) (1204)、 Pn (=①のノード) の dn (左) ノードに del の左ノード (③のノード) をセットする (626) (1205)。

し、②の図形情報の直後に入力された図形情報 (③の図形情報) の前には、②の直前に入力された図形情報 (この場合は存在しない。) が並ぶようにポインタをセットする (1211)。

以上で、図形情報の削除を行なう手順は終了する。

以上述べたように、ツリーよりのノード削除はノードごとに2分された枝を、1つのパスしか通らない。即ち、ツリー生成と同様の理由より、そのたどるノード数は全ノード数 (全図形情報数) を K とした場合、ほぼ $\log_2 K$ であることがわかる。

本システムの会話形図形編集プログラムは、操作者によって起動され、図形情報の編集指示を受けた場合、全図形情報の中から編集範囲の図形情報を前記ツリー探索処理で高速に検索し、グラフィック・ディスプレイに表示する。

なお、表示の際、図形情報が重なっており、描画順に表示図形が依存しているような場合は、検索されたノード・ポインタを適当な配列に保持し、

del の左ノード (③のノード) を Pn にセットし (627) (1206)、 dn に「左」を示すフラグ値をセットする (628)。 del の左、右ノードに Pn (=③のノード) の左、右ノード (ともに空ノード) をセット (629) (1207, 1208)、 Pn (=③のノード) の右ノードに n (=④のノード) をセットする (630)。

上述の処理操作を行なうことによりツリー・ノードの再構成を行なうことができる (1209)。この後、判定615へ戻るが、このときは del (=②のノード) の左、右ノードはともに空ノードとされているので (615)、 Pn (=③のノード) の dn (左) に空ノードをセットし (631)、ノードの削除処理を完了する (1210)。

そしてさらに、描画順序保持のためのポインタを削除する。即ち、②の図形情報の直前に入力された図形情報 (この場合は存在しない。) の次には、②の図形情報の直後に入力された図形情報 (③の図形情報) が並ぶようにポインタをセット

描画順にソートした後、表示する。更に、削除による表示画面の修復にも使用できる。

また、図形を描画するなどして、新たな図形情報の生成を行う場合は、新たな図形情報に基づいて外接する矩形座標を演算し、ノード生成処理を用いて図形情報の生成を行う。

既に生成されている図形情報を削除する場合は、適宜の手段で、指定された座標近傍の図形情報を、前記ツリー探索処理を行って、高速に検索する。

この場合は、指定された座標位置を含む小さな矩形 (たとえば、全空間を $[0, 1024] \times [0, 1024]$ とし、指定座標位置を x, y とした場合、 $[x-8, y-8] \times [x+8, y+8]$ で定義できる) を用いて、候補図形を検索することができる。

また、更に検索精度を上げるためには、複数の候補図形に対して指定座標点との間の距離計算を施し、優先順位を付けることもできる。

この実施例によれば、膨大な図形情報の度重なる検索をきわめて高速に行うことができ、操作者

に対して快適な編集作業環境を提供できる。

また、LBPやブロックに対し図形情報を出力する際に応用できることは明らかであり、膨大な図形情報の中の任意の範囲を高速に選択、出力することが可能である。

更に、図形情報を会話形式で入力せず、カードやフロッピーディスク等の媒体を通して、大量の図形情報を一括して入力させることも、ツリー生成時間（入力情報数を K とすると、ほぼ $K \log_2 K$ に比例した時間）が十分に短かいので、可能である。

また、第3図に示すように、2次記憶装置21の容量が不足しており、ツリーを格納できない場合でも、会話編集開始時に主記憶上で、ツリー生成プログラム22によってツリーを構成し、編集操作時の応答性を確保することもできる。

本実施例で説明したツリーの構造は、矩形の x 座標に着目したものであるが、全図形情報の座標系が $[0, N] \times [0, M]$ であり、 $M \gg N$ の場合は、明らかに、 y 座標に着目してツリーを

構成すべきである。このようにすれば、ツリーのレベルを低く押えることができる。また、座標系 $[0, N] \times [0, M]$ を任意の実数空間に拡張することも可能である。

（発明の効果）

以上述べたところから明らかなように、座標をキーとして図形情報を検索し、編集出力させる方式に、バイナリ・ツリーを導入したことにより、従来では不可能であった高速応答が可能となった。

すなわち、従来の応答時間を kK とした場合、 $(k+k') \log_2 K$ に短縮することができる。ただし、上記においては、 K は図形情報数、 k は検索処理のオーバーヘッド、 k' はツリー探索のオーバーヘッドである。

さらに、検索が高速化され、検索に要する時間が短縮されたことにより、大量の図形情報を一元的に管理することが可能となり、その任意範囲の編集、出力が従来技術にて確保されていた応答時間とほぼ変わりなく確保できることにより、より高度の操作性を持つことができる。

結果的に、グラフィックス・システムとしての性能が、飛躍的に向上することになる。

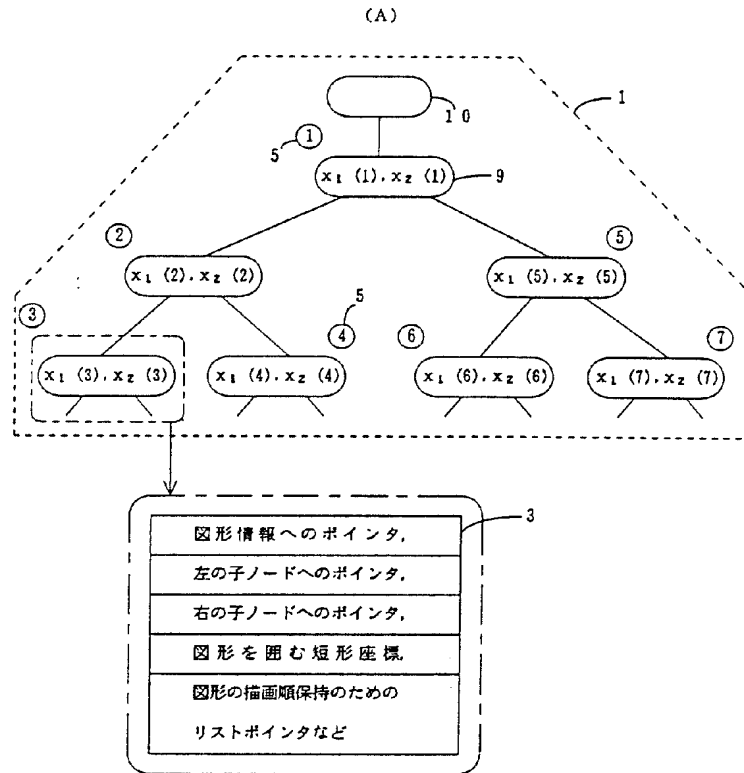
4. 図面の簡単な説明

第1図は本発明の要部であるバイナリ・ツリーの構造と図形情報の概要を示す図、第2、3図は本発明を適用したグラフィックス・システムの概略構成例を示すブロック図、第4図は本発明におけるツリーの探索処理を示すフローチャート、第5図はツリーの生成処理を示すフローチャート、第6図はツリーよりノードを削除する処理を示すフローチャート、第7図はバイナリ・ツリーの新規作成時におけるツリー構造の変化状態を示す図、第8～10図は図面情報の追加に伴うバイナリ・ツリーのノード追加時におけるツリー構造の変化状態を示す図、第11図はバイナリ・ツリーの探索経路を示す図、第12図は図面情報の削除に伴うバイナリ・ツリーのノード削除時におけるツリー構造の変化状態を示す図、第13図はバイナリ・ツリーを用いた図形情報の探索手順を説明するための座標系を示す図である。

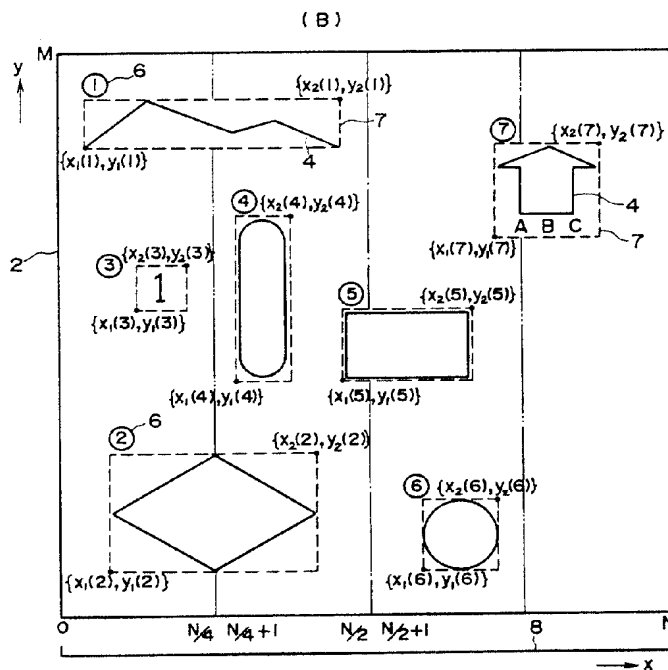
1…ツリーの全体構成、2…図形情報の全体構成、3…ツリー・ノード、4…図形情報、7…図形情報を囲む外接矩形、8…ルート・ノードに対応した区間 $[0, N]$ 、11…グラフィック・ディスプレイ、12…キーボード、13…電子計算機、14…マウス、15…ライトペン、16…デジタイザ、17、21…2次記憶装置、18…LBP、19…ブロック、20…バッチ型入力装置、22…ツリー生成プログラム

代理人 弁理士 平 木 道 人

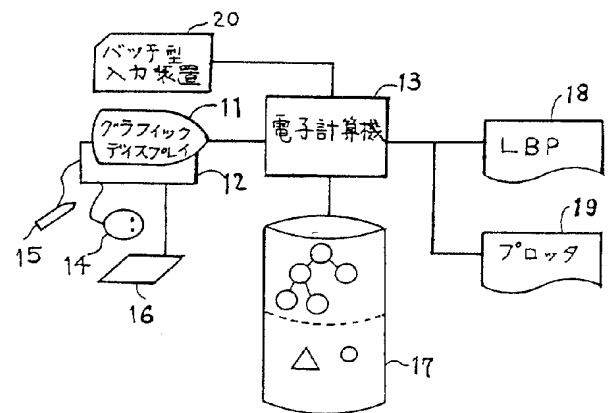
第 1 図 (その1)



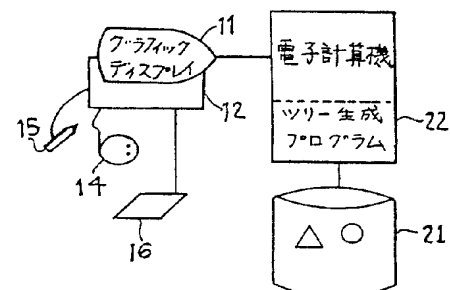
第 1 図 (その2)



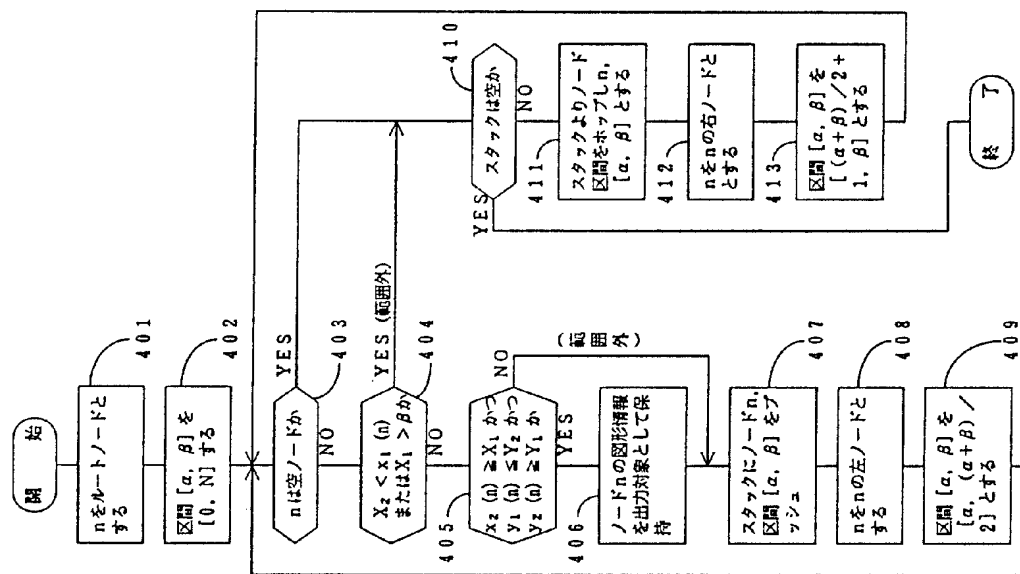
第 2 図



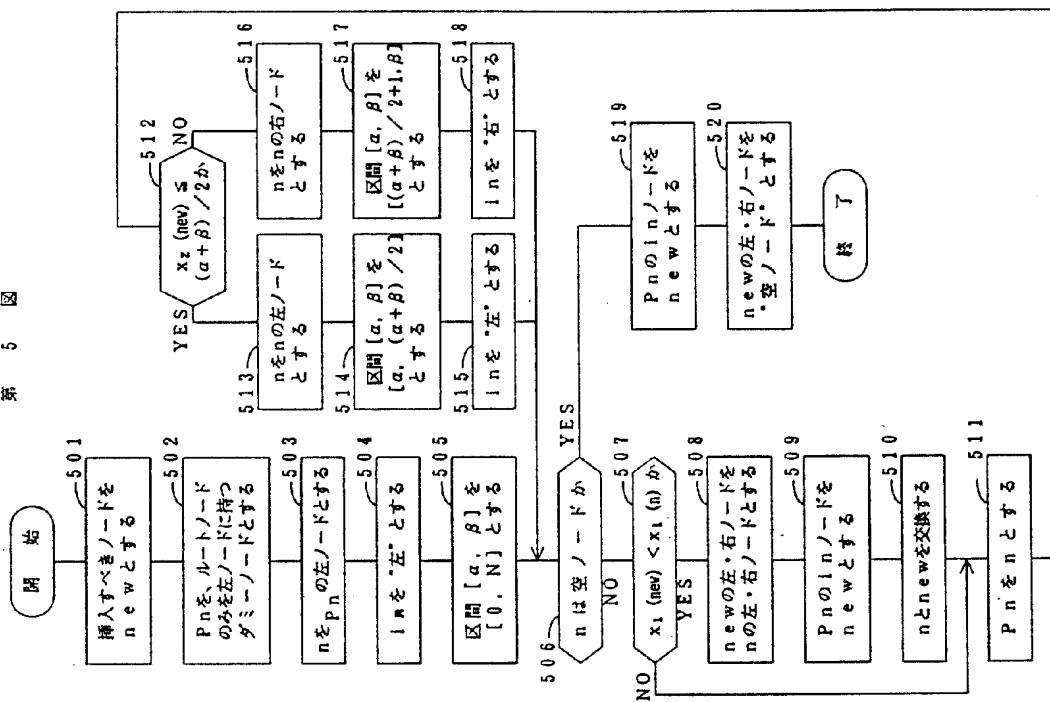
第 3 図



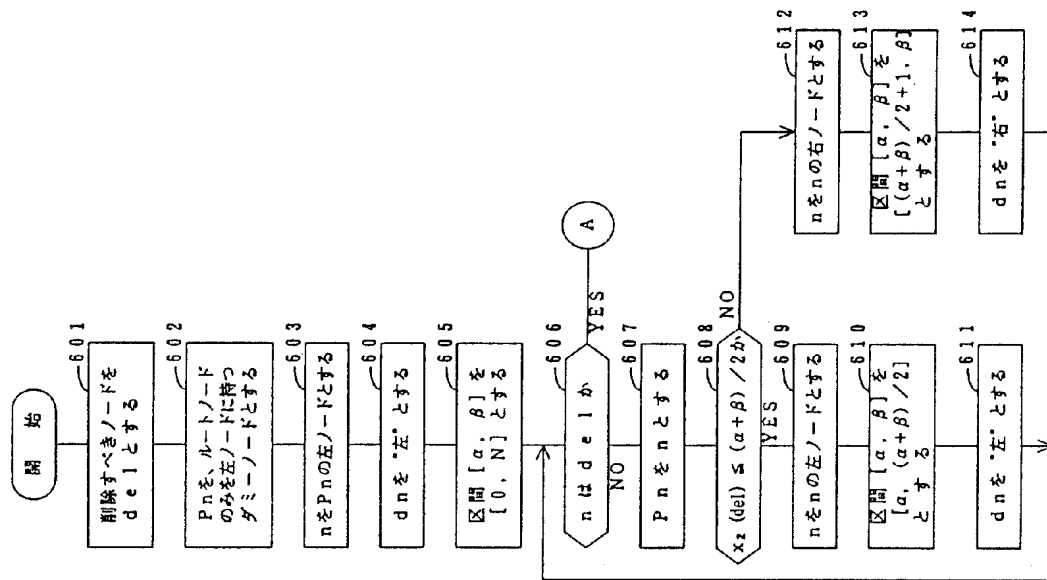
第 4 図



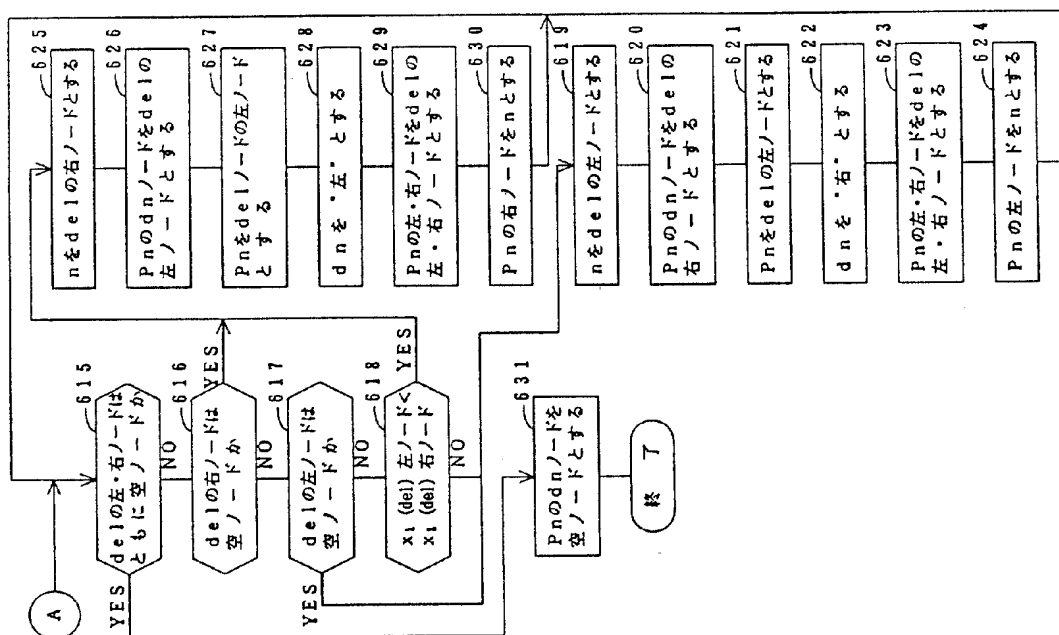
第 5 図



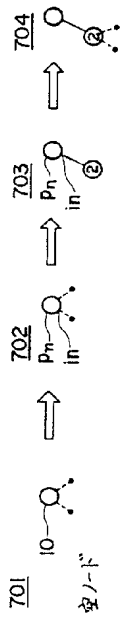
第 6 図



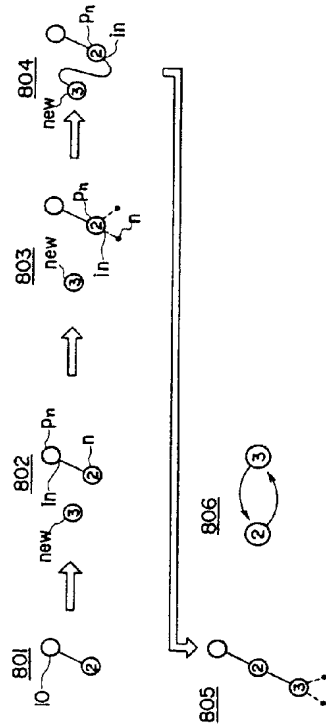
第 6 図



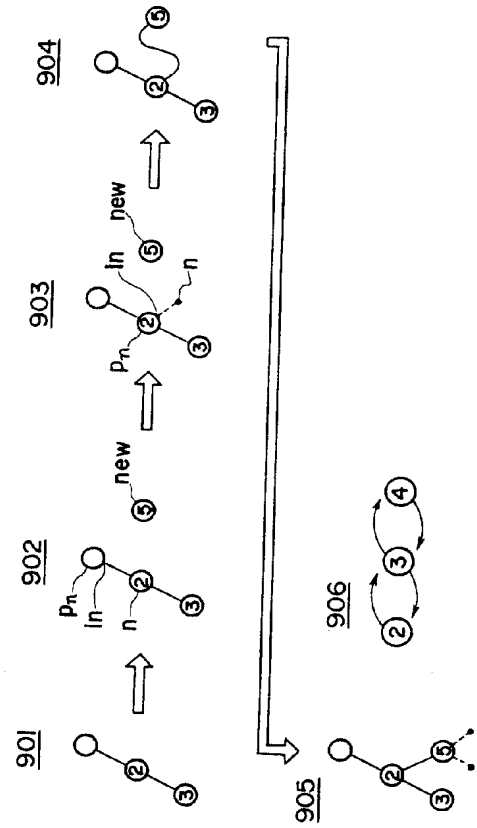
第7図



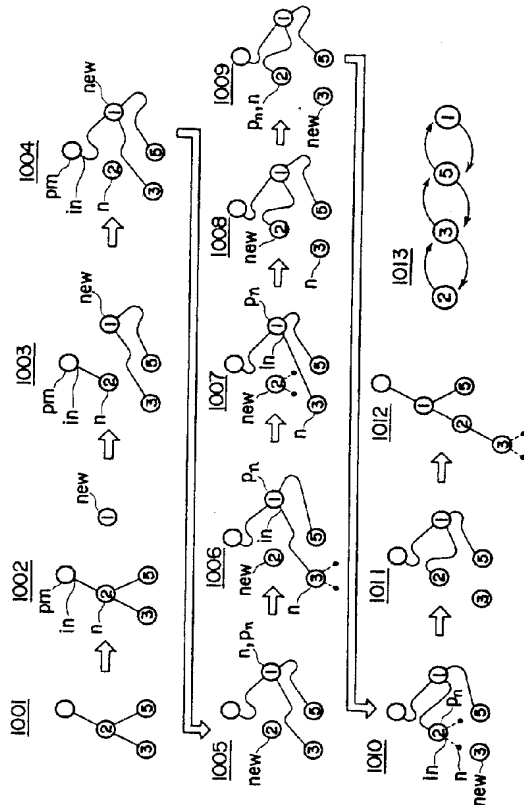
第8図



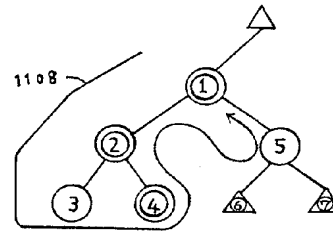
第9図



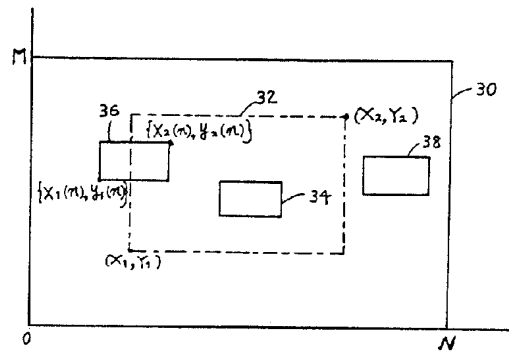
第10図



第11図



第13図



第12図

